



日 本 国 特 許 庁
JAPAN PATENT OFFICE

別紙添付の書類に記載されている事項は下記の出願書類に記載されている事項と同一であることを証明する。

This is to certify that the annexed is a true copy of the following application as filed with this Office.

出 願 年 月 日 2 0 0 2 年 1 2 月 6 日
Date of Application:

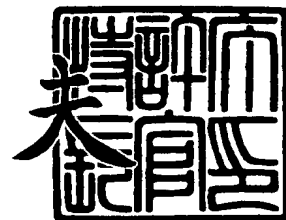
出 願 番 号 特 願 2 0 0 2 - 3 8 3 0 8 3
Application Number:
[ST. 10/C]: [J P 2 0 0 2 - 3 8 3 0 8 3]

出 願 人 加 藤 宝 一
Applicant(s):

2 0 0 3 年 1 2 月 4 日

特許庁長官
Commissioner,
Japan Patent Office

今 井 康



【書類名】 特許願

【整理番号】 KTMEPS21

【提出日】 平成14年12月 5日

【あて先】 特許庁長官 殿

【発明者】

【住所又は居所】 大阪府大阪市鶴見区鶴見 5 丁目 6 番 9 号 3 0 1 室

【氏名】 加藤 宝一

【特許出願人】

【住所又は居所】 大阪府大阪市鶴見区鶴見 5 丁目 6 番 9 号 3 0 1 室

【氏名又は名称】 加藤 宝一

【提出物件の目録】

【物件名】 明細書 1

【物件名】 図 面 1

【物件名】 要約書 1

【書類名】 明細書

【発明の名称】 簡易式入出力プログラミングシステム

【特許請求の範囲】

【請求項 1】 信号入出力及びアナログ入出力及び操作キー入力及び通信送受及び表示出力のプログラミング作製のため、特定言語へのマッピングを行う、選択型コマンドジェネレーター。

【請求項 2】 請求項 1 で作製された言語の多重処理用簡易コンパイラー。

【請求項 3】 請求項 2 のコンパイラーでコンパイルしたオブジェクトを実行させるコントローラー。

【発明の詳細な説明】

【0001】

信号入出力処理、アナログ入出力処理、通信送受処理、操作キー入力処理、表示出力処理を初心者が平易に設計し得る旨のビジュアルかつ記述選択型のコマンドジェネレーターをパソコンにて実行させ、コーディング手法を用いずとも、システム運用を遂行させるものである。

【発明の属する技術分野】

本発明は、産業用計測機、制御機のプログラミング手法の平易化に関する。

【0002】

【従来の技術】

コンピューター一般における従来のプログラミング手法は、コーディング形式によるもの。産業用入出力処理装置としては線画及び記号を付加して行くピジュアル形式のラダー図によるものが主たるものである。コーディング形式は近年、より一層拡充された各種コンパイラーがそれぞれの機能を競っており、優劣をつけがたいものがあるが、概して、元来コンパイラーは関数演算、ファイルマネジメント、画面入出力等諸機能を時系列で処理するを旨とし、時には多重処理用 OS を持ったものもある。しかし、多点の I/O を処理するにはコーディング文字数が多くなり、不便である。一方ラダー図の方はハードウェア技術者にとって設計回路に近く、ラダー図シーケンシャル処理は先頭ステップから末尾ステップまでを一瞬にして実行をサイクリックに繰り返すもので I/O 専一機としては高速

であるが、時系列のヒストリー組立てには熟練度を要する。つまり統合する意志的なものは常にユーザー側にあり、個人的な個性、能力差によってはシステム状態数の増大にともない、幾何級数的に複雑度が増大する。更にプログラムの流れは一次元的に流れるので、多重処理用OSを持たないものは、フラッグやメモリスイッチやメモリーのスケジューラーを製作者が作り出して多重処理を構築する。

【0003】

【発明が解決しようとする課題】

以上のように述べた従来のプログラミング製作システムでは初心技術者では状態ステップが多ければ多いほど開発出来なくなる（あるいは長期間になる）ので、この問題を一挙に解決するものである。

【0004】

問題解決に3通りあり、第一に、コーディングミスをさけるためにも、ネーミング検証方式にて処理し、記述選択方式にて、コーディングさせない事。

第二に、多重処理と各処理の連携の統合、規制をコマンドジェネレーターもしくは簡易コンパイラーで統合、規制すべきものである事。

第三に、多重処理を意識せずに各々のタスクについてはあくまでもフローチャートの流れを反影出来る形式でなければならない。極論すれば命令書の参照なしに製作出来る事。音声会話形式のロボットティーチング指令の将来へつなげる事。

【課題を解決するための手段】

本発明は上記目的を達成するため、コントローラーにおける多重OS用のカーネル（核）を必要最小なものとして、各々のタスクの現在状況を示す、フラッグとイベント発生を待つエントリーアドレス、その時のスタックポインター、各々タスクが持つタイマーカウンター等々をサーチするモニター、タイマーチェック割り込みにおけるタイマーのカウントダウン処理に留める。

【0005】

簡易コンパイラー言語に入出力内部信号待ちと監視タイマーと組み合わせて1パックにした、1ステップ動作と動作の完全遂行をチェックするマクロ命令を製

作する。

更に、必要箇所で遅延及びイベントウェイトを行うマクロ命令を作る事により、フローチャート形式にする。

更に一次元的に一瞬にしてサーチするタスク行を並べ、それぞれのタスク内の動作をコマンド列にしフローチャート形式で書ける完全二次元的構造にする。

更に二次元表現のため、従来のコマンドステートメントの単位が改行ベースなのに対して、本コンパイラーのコマンドの区切りにスペースを使用、右端で改行となるまでコマンドステートメントを横に展開出来るようにする。その事により、タスク行コマンド列の二次元表現とする。コメントは；から；まで又は改行までをコメントとしているからどの箇所でも挿入出来る。

更に他のコンパイラー同様各種関数を用意する。

【0006】

コマンドジェネレーターとして、各種処理のメニュー選択をさせる。

更にタスク列を必ず宣言するようにコマンドジェネレーターで製作はガイダンスに基づき、ミスを防ぐ事。

更にコマンドジェネレーターで入出力内部信号の1点1点をネーミングさせ、注釈も表示させ、ミスを防ぐ事。

更にコマンドジェネレーターを使用せず、直接簡易コンパイラーを修正して、熟練度に応じて更なる応用性を拡充出来るようにすべくエディターも内包する事。

更にコマンドジェネレーターはパソコンとコントローラーをRS232Cで接続して、プログラム転送及びデバッグ機能を有する事。

【0007】

【発明実施の形態】

以下、本発明の実施の形態を図1～図50に基づいて説明する。

図1の如く、パソコンPCAT互換機WINDOWSにて、コマンドジェネレーターで生成されたマクロ言語を簡易コンパイラーでコンパイルし、コントローラーへオブジェクトをRS232Cで転送して実行するものである。

更に簡易コンパイラーで生成されたマクロ言語はテキスト形式なので、コマン

ドジェネレーターなしでも、コンパイルして転送できるものである。したがってコマンドジェネレーター、簡易コンパイラー、コントローラーと分けて説明する。

【コマンドジェネレーター】

【0008】

コマンドジェネレーターは図1の如く、コマンドジェネレーター、それに併う独自開発のエディター、デバッガーを内包する。コマンドジェネレーターはコマンドメニューにて実行する。エディターは編集メニュー、デバッガーはデバッグメニューにて実行する。

エディターは、コピー、切り取り、貼り付け、検索、置換機能を持つ。

デバッガーは、デバッグモードでの実行（コントローラーRAM上で実行）、カーソルの前まで実行（ブレイクポイント）、ステップ等の機能を持つ。

【0009】

図2は、コマンドジェネレーターの基本画面である。

図中、メニューの信号をクリックして入力、出力、内部信号のアサインを行う。これは端子番号にネーミングとコメントを入れる事で、システムの設計と確認を容易にするためである。マイクロソフトコンボボックスで信号ネームを、ツールチップテキストを用いてマウスの変化でコメントを表示するようにした。図3は入力信号、以下出力、内部信号は同様である。

【00010】

図2におけるコマンドメニューで選択をした画面で代表的なものとして信号待ちを表示したものが図4である。タイマー付き入力待ち選択をして、入力信号待ちコマンドステートメントを生成させる。そして入力信号待ちに関するフローが図5である。これだけが簡易コンパイラーの一つのコマンドステートメントに相当する。

更に図2においてコマンドメニューでステップ動作を選択すれば図6の画面になり、ステップ動作の簡易コンパイラーへの生成は複数のコマンドステートメントとなる。図7中、*1印はオプションによるもので、最小2最大5コマンドステートメントを生成する。このコマンドステートメント列の連続で大体のシーケ

ンス処理は完行される。図5の監視タイマーのセットは必ずしも必要でなく、その前にタイマーセットされており、信号チェックをひとまとめに監視するような場合、継続タイマーの選択も出来る。

【0011】

純動作処理の他、制御データの packets 毎の搬送のため図8の如きシフトメモリー動作コマンドを用意して、回転テーブル、ライン搬送の実際的なデータフローに適應させる。動作イベントは入力、出力、内部信号の立ち上がり、立ち下がりで行う。

【0012】

プロフラムステップ短縮する目的で縦×横のデータテーブルを用意し、インデックス的にデータを獲得する。図9、これにて例えば、テーブル化されたスピードコントロールデータ出力等の処理を簡易的に作成できる。

【0013】

更に図10の如くAND, ORゲートの選択が出来、信号判定と条件ジャンプ文を使用せずとも、適應したロジックを作製出来る。

【0014】

更に図11の如く4×4要素(図は4×3)のラダー論理図で信号のAND, ORを連結クリックと登録クリックで行い、従来の線画エディットに比して、快速に回路を作製出来る。

【0015】

更に図12の如くフリッカー選択で、アラーム等のフリッカーのコマンド列を選択可能にする。

【0016】

更にレジスター関係の演算命令、関数、送受信マクロ、日付時計読み出し、アナログ入出力命令、タイマー命令等も記述型選択方式にてコーディング無しでコードを生成する。(図13, 図14, 図15, 図16, 図17, 図18)

【0017】

更にマルチタスク関連の命令は図19の如くまとめ、簡単にタスク間の連結を行う。但しデータのやりとりは行わない。本システムのデータ、レジスター、信

号は全てパブリック状態にある。

【0018】

更にコントローラー部のキー入力及びLCD出力の簡易化にともなう命令を図20の如くまとめた。

【0019】

更にジャンプ系列の命令、入出力内部信号の判定、セット命令も選択方式にて平易化した。(図21, 図22)

【0020】

コマンドジェネレーターは図1の如く、コマンドジェネレーター又は簡易コンパイラーの直接エディットを、用いて作製された言語をコンパイルしてオブジェクト生成後、コントローラーに送信して実行させるものである。デバッグモードまでは本コマンドジェネレーターの管理下にあるが、デバッグ完了後は、RS232Cを切り離し、コントローラ単独でフラッシュメモリー書き込み、作製されたプログラムを実行するものである。

【0021】

本コマンドジェネレーターに要するシステムはマイクロソフト各種ツール、コンポーネントである。

【簡易コンパイラー】

【0022】

本コンパイラーは多重処理用に独自に開発したものである。

更に開発効率を高めるため、コンパイラー本来の特性、冗長度を持たした言語的な特性から、コンパクトに表現をする様、心がけたものである。そのため、ネモニックの工夫と命令の区切りをスペースにする等した。重点を入出力制御におき、一般のコンパイラーの様に演算式が自由でない。更に配列、インデックス等の機能は持たない。

更に構造上のクラス、メンバー等といった概念も持たない。但し、入出力の監視タイマー付取込み命令等1命令として特殊なものもある。(図5)

更にタスク命令として、生成、消滅、休止、リセット、ステータス等の命令を使用して、多重処理を実現させる。

【0023】

本コンパイラーは“=”と“_”（アンダーライン）を中心として命令解析をツリー構造で行うものである。“=”は右辺の式、又はデータを左辺へ演算、又は変換、又は直接代入するものである。

更に“_”（アンダーライン）は左部のコマンドに対して、右部の詳細仕様に基づいたコマンドを実行する。命令にはタスク命令、ジャンプ命令、タイマー命令、A/D、D/A、送信、受信命令、演算命令、キー入力LCD表示命令がある。

【0024】

命令ネモニックの中でタスク命令以外でも“S”文字や“W”文字が含まれている命令はタスクを切替（スイッチング）します。特に“W”含む命令は、その所（アドレス）で、とどまって、イベントをウェイトしている状態である。

更に“S”は送信を意味し送信完了まで、その所（アドレス）でとどまってウェイトする。

更に“%”は受信を意味し受信完了までウェイトする。

更に命令ネモニックの中に“?”文字が含まれている命令は、判定true（1）かfalse（0）をチェックするものである。したがって次の命令は条件ジャンプが必ず必要となる。

【0025】

各命令一覧表を図23、図24、図25、図26、図27、図28、図29、図30、に記す。便宜上タスクラベル名をTaskn、ラベル名をPclabel、レジスターNOを10、20、30、データを9の複数桁又は最大値、入力信号をIsig、出力信号をOsig、内部信号名をVsig、アスキー文字列を“ABC”、16進文字列をh’31、32、33、として表現する。

【コントローラー】**【0026】**

コントローラー仕様は図1如くである。更にコントローラーメモリー空間上、常駐モニター及び常駐サブルーチンを有する。

【0027】

PCから受信したコンパイル後のユーザーオブジェクトプログラムをそのままRAM上にデバッガーモードで走らせる事が出来る。

更に確認後、フラッシュメモリーへ書き込み処理を行い、PCと切離してコントローラー単独で実用処理機となる。

更に、常駐のサービスプログラムを有する。PCから転送されたユーザープログラムが開発者独自の自動プログラムとすれば、このサービスプログラムは手動処理的に提供しているものである。このサービスプログラムは主に3形態を有し、1つにはコントローラーのPCとの受信処理、フラッシュメモリー書き込み等のコントロール部と、1つには通信条件等のパラメータ設定、カレンダークロックの設定等の設定部と、1つには、I/O入出力、A/D、D/A入出力RS232C又はRS422の送信受信、各々の手動チェック部とがある。そしてこれらの選択は初期画面から各々ツリー構造をなす画面推移で実現出来るようにしている。

【0028】

コントローラーの仕様はI/Oフォトカプラー付き32点/32点、A/D 12ビット×8CH 0-4.095V、D/A 12ビット×2CH 0-5V、カレンダー時計、通信RS232C 1CH、RS422 1CH、128×64ドット LCD（横16桁×縦6桁アスキー文字表示器）、12個のキー、512KBフラッシュメモリー、512KBスタティックRAM（バッテリー付）CPU Z8010MHzである。

【0029】

内部メモリーとして本コマンドジェネレーター、本簡易コンパイラー、本コントローラー共通の設計仕様として、

- v00-vFF、 256点の内部接点
- i00-i99、 100個の16ビットレジスター
- d00-d99、 100個の16ビット定数格納レジスター
- t00-t99、 100個の16ビットタイマー定数レジスター
- r00-r99、 100個の32ビット実数レジスター
- c00-c99、 100個の8ビット文字レジスター

Tm0-Tm9、 10個の16ビット外部タイマー

各タスク1個16ビット固有タイマー

最大タスク数120

システムサポート 内部接点 v f f 1 m s デューティクロックサービス

内部接点 v f e 1 0 m s デューティクロックサービス

内部接点 v f d 1 s e c デューティクロックサービス

正整数レジスター i 9 9 m s カウンター (0-999)

電源ONのスタート時のデフォルト値として

実数レジスター r 9 9 = 0. 0

実数レジスター r 9 8 = 1. 0

実数レジスター r 9 7 = 1 0. 0

実数レジスター r 9 6 = 1 0 0. 0

実数レジスター r 9 5 = 1 0 0 0. 0

実数レジスター r 9 4 = 4 0 9 6. 0

実数レジスター r 9 3 = 3. 1 4 1 5 9

実数レジスター r 9 2 = 3 6 0. 0

実数レジスター r 9 1 = 2 7 3. 0

【0030】

12キー入力のアスキーコードとしての対応図は図31である。

【0031】

図32はメニュー画面、図33はキーの説明でコントロール部の各々の画面は図34～図36。

設定部の各々の画面は図37～図44。チェックの各々の画面として図45～図50である。

【発明の効果】

上述した本発明のコマンドジェネレーター、簡易コンパイラー、コントローラーで製作工数が極端に減少し、デバッグミスがほとんどなく、初心技術者にも短時間のインストラクト後、適確に目的とする動作をさせる事が出来る効果がある。

【図面の簡単な説明】

【図 1】

本簡易式入出力プログラミングシステムの構成図である。

【図 2】

PC 側コマンドジェネレーターの、メイン画面を示した図である。

【図 3】

コマンドジェネレーターの、入力信号アサイン画面を示した図である。

【図 4】

コマンドジェネレーターの、信号待ちコマンドの選択画面を示した図である。
。

【図 5】

上記、信号待ちコマンド中、入力信号待ちのフローチャートを示した図である。

【図 6】

コマンドジェネレーターの、ステップ動作コマンドの選択画面を示した図である。

【図 7】

上記、ステップ動作複合コマンドのフローチャートを示した図である。

【図 8】

コマンドジェネレーターの、シフトメモリーコマンドの選択画面を示した図である。

【図 9】

コマンドジェネレーターの、テーブル処理コマンドの製作画面を示した図である。

【図 10】

コマンドジェネレーターの、AND/OR コマンドの製作画面を示した図である。

【図 11】

コマンドジェネレーターの、ラダー図の製作画面を示した図である。

【図 12】

コマンドジェネレーターの、フリッカーコマンドの製作画面を示した図である。

【図 13】

コマンドジェネレーターの、レジスター操作コマンドの選択画面を示した図である。

【図 14】

コマンドジェネレーターの、Iレジスター操作コマンドの選択画面を示した図である。

【図 15】

コマンドジェネレーターの、Cレジスター操作コマンドの選択画面を示した図である。

【図 16】

コマンドジェネレーターの、その他の関数コマンドの選択画面を示した図である。

【図 17】

コマンドジェネレーターの、通信、アナログコマンドの選択画面を示した図である。

【図 18】

コマンドジェネレーターの、タイマーコマンドの選択画面を示した図である。

【図 19】

コマンドジェネレーターの、タスクコマンドの選択画面を示した図である。

【図 20】

コマンドジェネレーターの、キー入力、表示コマンドの選択画面を示した図である。

【図 21】

コマンドジェネレーターの、ジャンプコマンドの選択画面を示した図である。

【図 22】

コマンドジェネレーターの、信号入出力コマンドの選択画面を示した図である。

【図 2 3】

簡易コンパイラーの、タスク命令一覧を示した図である。

【図 2 4】

簡易コンパイラーの、ジャンプ命令一覧を示した図である。

【図 2 5】

簡易コンパイラーの、I/O命令一覧を示した図である。

【図 2 6】

簡易コンパイラーの、タイマー命令一覧を示した図である。

【図 2 7】

簡易コンパイラーの、A/D、D/A、送信、受信命令一覧を示した図である。

【図 2 8】

簡易コンパイラーの、演算命令一覧を示した図である。

【図 2 9】

簡易コンパイラーの、キー入力、LCD表示命令一覧を示した図である。

【図 3 0】

簡易コンパイラーの、その他の命令一覧を示した図である。

【図 3 1】

コントローラーの、キー配置とそのアスキーコード対応を示した図である。

【図 3 2】

コントローラーの、サービスメニュー画面を示した図である。

【図 3 3】

コントローラーの、キーとその役目、表示画面を示した図である。

(F 1, F 2 キーの意味はユーザーが決める)

【図 3 4】

コントローラーの、コントロールメニュー画面を示した図である。

【図 3 5】

コントロール部の、PCよりオブジェクトプログラムを受信する画面を示した図である。

【図 3 6】

コントロール部の、受信したオブジェクトプログラムをフラッシュメモリーへ書き込む画面を示した図である。

【図 3 7】

コントローラーの、設定メニュー画面を示した図である。

【図 3 8】

設定部の、データメニューにデータをセットする画面を示した図である。

【図 3 9】

設定部の、オーバータイマーにタイマー値をセットする画面を示した図である。

【図 4 0】

設定部の、通信 0 C Hのパラメーターをセットする画面を示した図である。

【図 4 1】

設定部の、通信 1 C Hのパラメーターをセットする画面を示した図である。

【図 4 2】

設定部の、通信 0 C H， 1 C Hのターミネーターと受信オーバータイムを設定する画面を示した図である。

【図 4 3】

設定部の、カレンダークロックを設定する画面を示した図である。

【図 4 4】

設定部の、A / D、D / A、オプション個数を設定する画面を示した図である。

【図 4 5】

コントローラーの、チェックメニュー画面を示した図である。

【図 4 6】

チェック部の、I / Oチェック画面を示した図である。

【図 4 7】

チェック部の、A／Dチェック画面を示した図である。

【図 4 8】

チェック部の、D／Aチェック画面を示した図である。

【図 4 9】

チェック部の、受信データ、0 C H， 1 C H画面を示した図である。

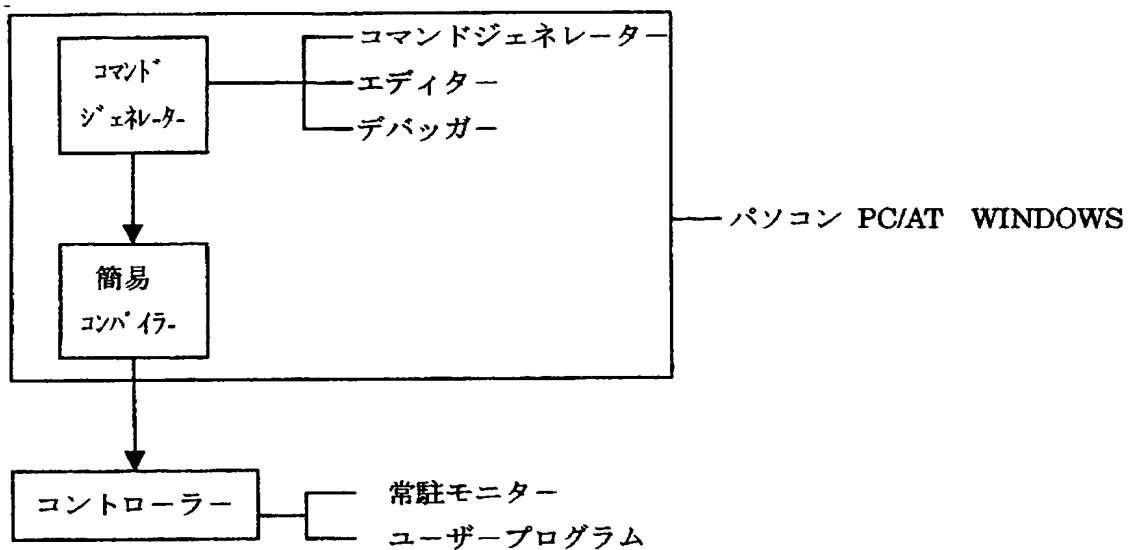
【図 5 0】

チェック部の、その他のチェック画面を示した図である。

(MAXサイクルタイム、I，C，Rレジスター内容、内部信号Vの内容)

【書類名】 図面

【図 1】



コントローラー仕様: 縦170mm x 横120mm x 高さ60mm
キー12入力、LCD 6×16桁、I/O各32点フォトカプラー付き、A/D 8CH、D/A 2CH、カレンダー時計、RS232C、RS422、CPU Z80 10 MHz、フラッシュメモリー512KB、スタティックRAM 512KB バッテリーバックアップ付き

【図 2】

Multi Easy Programming System						コマンド
ファイル	編集	信号	レジスタ	コントローラの条件	デバッグ	タスク制御
エディット フィールド						ジャンプ
						信号入出力
						信号待ち
						タイマー
						フリッカー
						通信 アナログ
						レジスター 操作
						I レジスター 操作
						C レジスター 操作
コマンド選択された場合の各種画面 (エディットフィールドのカーソル位置によっては 上部に展開する)						その他の 関数
						シフト メモリー
						テーブル 処理
						ステップ 動作
						AND/OR ゲート
						キー入力 表示

【図 3】

番号	設定	入力信号名	注釈	番号	設定	入力信号名	注釈
0				16			
1				17			
2				18			
3				19			
4				20			
5				21			
6				22			
7				23			
8				24			
9				25			
10				26			
11				27			
12				28			
13				29			
14				30			
15				31			

【図 4】

ラベル

☐ 入力信号待ち の ☐ ON を待ちます。
☐ 出力信号待ち の ☐ OFF を待ちます。
☐ 内部信号待ち の

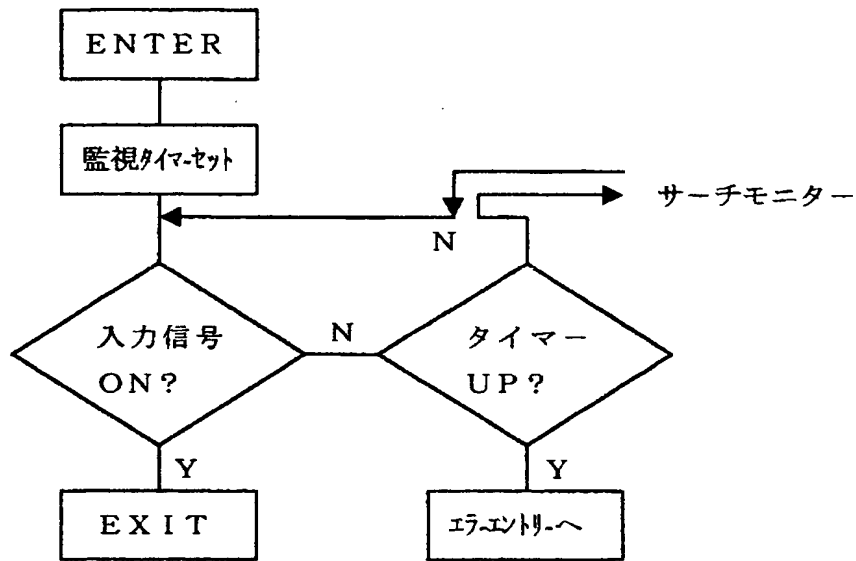
但し

☐ 無条件に待ちます。
☐ タイマー設定し、監視状態にします。
☐ 監視状態は以前設定のタイマーのまま継続させます。

タイムオーバーなら、
ラベル に
ジャンプします。
-----> エラー処理

オーバータイマー NO

【図 5】



【図 6】

ラベル

☐ オーバertimeチェック無し
☐ オーバertimeチェック有り
☐ オーバertimeチェック継続
 オーバertime-NO

信号 の ☐ I ☐ V

☐ 遅延をおかず
☐ 遅延をおいて

遅延 * 10ms

信号 を

☐ ON する
☐ OFF する
☐ ON して遅延後 OFF
☐ OFF して遅延後 ON
 遅延 * 10ms

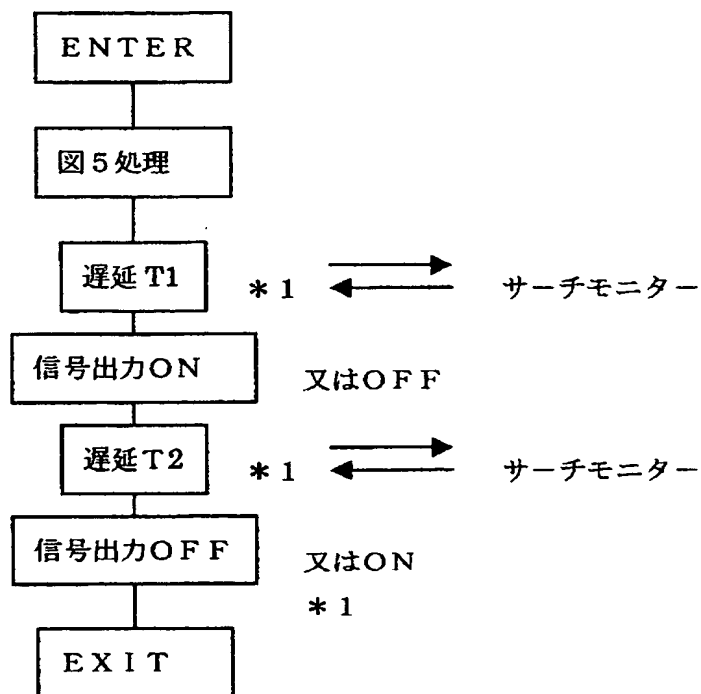
☐ 立ち上がり
☐ 立ち下がり

エラ-処理ラベル

オバ-?
 yes → エラ-処理ラベル
 no →

挿入 上書 削除

【図 7】



【図 8】

ラベル

メモリーの種類は ☐ 文字 CXX ☐ 正数 0-65535 IXX で、 XX のメモリーを、

信号 の ☐ 立ち上がり ☐ 立ち下がり で ☐ 遅延をおかず ☐ 遅延をおいて

☐ I ☐ V ☐ O 遅延タイマNO

yy

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

yy

の先頭メモリーから 個分(1-15)を、1ピッチ

右にシフトすると同時に XX のメモリーを先頭の位置にセットします。参照は、I、C、それぞれメモリー参照で行う。

【図 9】

ラベル

☐ テーブル作成

☐ バイトテーブル
☐ ワードテーブル

縦 × 横 のテーブルで
 入力方法は、16進で入れて下さい。
 テーブルのラベルは、必ず入れて下さい。

Iレジスタのポインター

	0	1	2	3	4	5	6	7	8	9
0										
1										
2										
3										
4										
5										
6										
7										
8										
9										

ページ送り

ページ戻り

☐ バイトテーブル読み込み

☐ ワードテーブル読み込み

I で、ラベル のテーブルから、横

↑

テーブルの縦ポインター

↑

テーブルの横列と同じにする

バイト分、C

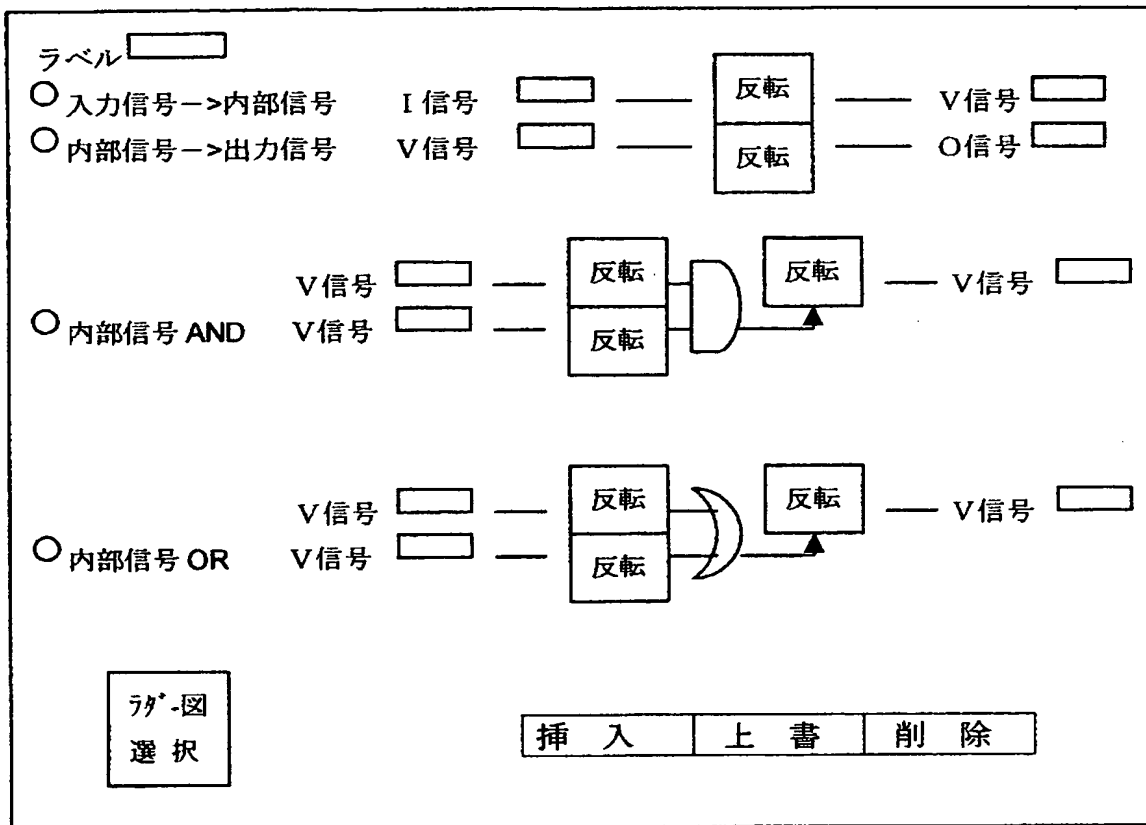
ワード分、I の先頭からセットする。

挿入

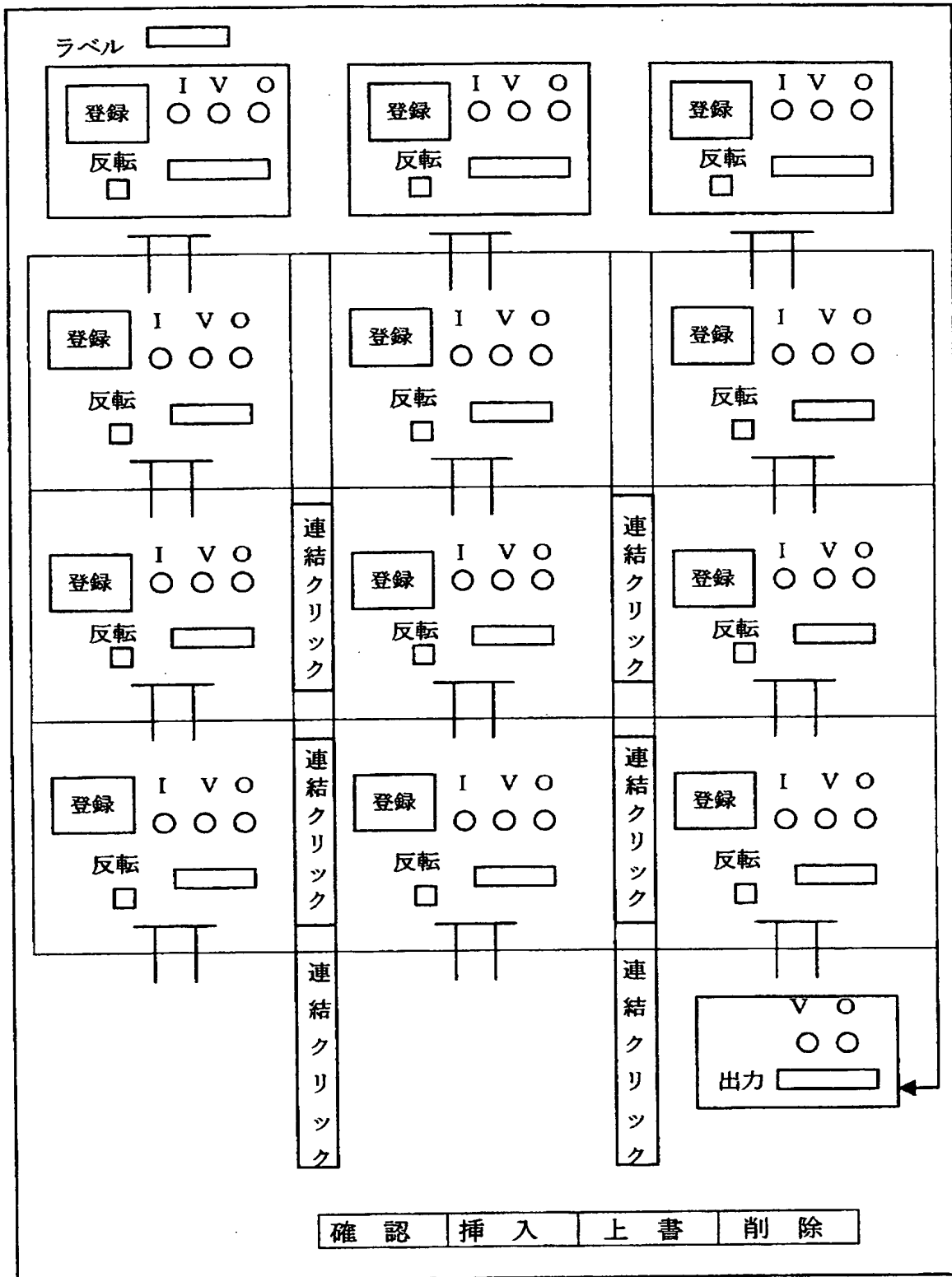
上書

削除

【図 10】



【図 11】



【図 12】

ラベル 必ず入れて下さい。

☐ 入力信号 ON で出力信号 を 間隔 X10ms
☐ 内部信号

間隔 X10ms でフリッカーします。

【図 13】

ラベル

☐ I 正整数レジスタ-同志の演算式 $I \text{ } = I \text{ } \begin{matrix} \text{+} \\ \text{-} \\ \text{*} \\ \text{/} \end{matrix}$

☐ I 正整数レジスタ-と定数の演算式 $I \text{ } = I \text{ } \begin{matrix} \text{+} \\ \text{-} \\ \text{*} \\ \text{/} \end{matrix}$

☐ R 実数レジスタ-同志の演算式 $R \text{ } = R \text{ } \begin{matrix} \text{+} \\ \text{-} \\ \text{*} \\ \text{/} \end{matrix}$

☐ I 正整数レジスタ-への定数の算入 $I \text{ } = \text{}$
☐ I 正整数レジスタ-から R 実数レジスタ-への変換 $R \text{ } = I \text{ }$
☐ R 実数レジスタ-から I 正整数レジスタ-への変換 $I \text{ } = R \text{ }$

【図 14】

ラベル

☐ I 正整数レジスタ-同志の論理式 $I \text{ } = I \text{ } \begin{cases} \text{AND} \\ \text{OR} \end{cases} I \text{ }$

☐ I 正整数レジスタ-と定数の論理式 $I \text{ } = I \text{ } \begin{cases} \text{AND} \\ \text{OR} \end{cases} \text{ }$

☐ I 正整数レジスタ-の左回転 $I \text{ } \leftarrow \text{ } (1-15)\text{回 1ビットずつ左回転して}$

☐ I 正整数レジスタ-の右回転 $\text{ } \rightarrow I \text{ } (1-15)\text{回 1ビットずつ右回転して}$

☐ I 正整数レジスタ-の左シフト $\leftarrow I \text{ } (1-15)\text{回 1ビットずつ左シフトして}$

☐ I 正整数レジスタ-の右シフト $I \text{ } \rightarrow \text{ } (1-15)\text{回 1ビットずつ右シフトして}$

I にセットする。

☐ I 正整数レジスタ-の 10 進変換 2 進数 I を 10 進数 $I \text{ }$ に変換する。

☐ I 正整数レジスタ-の 2 進変換 10 進数 I を 2 進数 但し 0 から 9999 の範囲。

☐ I 正整数レジスタ-をインデックスとした、データメモリーへの書き込み
I をインデックスとして、I の内容を、データメモリーに書き込む。

☐ I 正整数レジスタ-をインデックスとした、データメモリーからの読み込み
I をインデックスとして、I へ、データメモリーから読み込む。

挿入

上書

削除

【図 15】

ラベル

☐ 文字レジスタ-直接入力 C の先頭から ☐ アスキーコード を文字列とする。
☐ 16進コード

☐ Iレジスタ-から文字レジスタ-入力 I の内容、2進数 ☐ 16進で4桁
☐ ゼロパディング無しでの10進で 桁
☐ ゼロパディング有りでの10進で 桁

として、C の先頭からの文字列とする。

☐ Rレジスタ-から文字レジスタ-入力 R の内容 ☐ ゼロパディング無しでの10進で
☐ ゼロパディング有りでの10進で

正数 桁、小数 桁を C の先頭からの文字列とする。

☐ 文字レジスタ-から Iレジスタ-入力 C を先頭とした文字列

☐ 16進で4桁
☐ ゼロパディング有り無しでの10進で 桁

を I へ2進化入力する。

☐ 文字レジスタ-から Rレジスタ-入力 C を先頭とした文字列 桁を R へ
 浮動小数点化入力する。

挿入 上書 削除

【図 16】

ラベル

☐ 平方根 $R \text{ } = \text{SQRT} (R \text{ })$
☐ A の X 乗 $R \text{ } = (R \text{ }) ** (R \text{ })$

☐ Sin $R \text{ } = \text{SIN} (R \text{ })$
☐ Cos $R \text{ } = \text{COS} (R \text{ })$

☐ Arc Tan $R \text{ } = \text{ATN} (R \text{ })$
☐ 自然対数 $R \text{ } = \text{LN} (R \text{ })$

☐ 常用対数 $R \text{ } = \text{LOG} (R \text{ })$

【図 17】

ラベル

☐ 送信 文字レジスタ- C の先頭から バイト分送信

☐ 受信 受信を待ち、受信があれば、受信バッファ-から、文字レジスタ- C の先頭へ、先頭から バイト分セットします。

☐ アナログ入力 (0-7) ch から I へ入力

☐ アナログ出力 (0-1) ch へ I から出力

☐ 時計入力 文字レジスタ- C の先頭から、12 バイト分(年月日時分秒) セット

【図 18】

ラベル

☐ タイマセット グローバルタイマ- (0-9) に、タイマ設定 NO (2桁) の内容値(x10ms) をセットする。

☐ タイマチェック グローバルタイマ- (0-9) が、0 か、0 でないかをチェックします。結果を条件ジャンプで判断します。

☐ 固定遅延 (0-65535 x 10ms) の定数だけ、本命令の所で遅延します。

☐ 変数遅延 正整数レジスタ- I の内容(0-65535 x 10ms) だけ、本命令の所で遅延します。

☐ タイマ-NO 遅延 タイマ-NO の内容(0-65535 x 10ms) だけ、本命令の所で遅延します

挿入	上書	削除
----	----	----

【図 19】

ラベル

☐ タスク登録 (タスクの始まりです。ラベルを必ず設定のこと。) コメント

☐ 他タスク起動 (ラベル のタスクを起動する。)

☐ 他タスク休止 (ラベル のタスクを休止する。中途再開は、他タスク起動。)

☐ 他タスクリセット (ラベル のタスクをリセットする。他タスク起動で、先頭から実行。)

☐ 他タスクチェック (ラベル のタスクが起動状態にあれば、True(1)、そうでなければ、False(0)になります。)

☐ タスクスイッチ (自タスクからループ形式の次タスクへ、制御権を移行させる。)

☐ 自タスクリセット (自タスクを消滅終結。)

挿入	上書	削除
----	----	----

【図 20】

ラベル

☐ キーステータス ☐ I ☐ へ即入力 ☐ キーコード入力待ち ☐ C ☐ へ押下後入力

☐ コード範囲チェック ☐ C ☐ のコードが MIN<==<=MAX の ex, "0," "9" or h '30, h' 39
 範囲に無ければ、ラベル へジャンプします。

☐ 画面クリア

☐ 固定メッセージ表示 ☐ X ☐ Y ☐ から、7ビットコード列 ☐ を表示します。ex, "ABCDEFGHJK"

☐ 反転表示 ☐ X ☐ Y ☐ から、☐ 文字分、反転表示します。

☐ 数値入力表示 ☐ X ☐ Y ☐ から、☐ 文字分、数値入力表示しながら、エントリで、
☐ C ☐ を先頭として、メモリセットします。クリアの場合は、true(not 0)となって、
 ラベル へジャンプします。エントリはジャンプしないで、 16 進入力
 続行します。

☐ 反転表示ビットインターセット ☐ I ☐ で示す反転ビットインターのセット。
 各々 X,Y,length をセットし、最後のカマは無し。

☐ 反転表示ビットインターインクリメント I で示す反転ビットインターのインクリメント。
 上記と同じデータをセットする。

【図 21】

ラベル

☐ 無条件に _____ ラベル へジャンプします。

☐ False(0)なら _____

☐ True(0 でない)なら _____

☐ もしも(IF) _____

☐ I が

☐ I
☐ 定数

☐ R が R より

☐ >=ならば
☐ <=ならば
☐ > ならば
☐ < ならば
☐ =ならば
☐ <>ならば

ラベル へジャンプします。それ以外は続行します。

☐ 選択 文字レジスタ- C のコードによるジャンプ ex、"A",Abc1, h' 42, Abc2
 (コードとラベルをペアとして、カンマで区切ること。)

☐ 無条件に _____ ラベル へコールします。

☐ サブコールの終了。コールしたところの次のところへ戻る。

挿入

上書

削除

【図22】

ラベル

○ 入力信号 ☐ を入力して

○ 内部信号 ☐ をチェックして

○ 出力信号 ☐ をチェックして

○ 出力信号 ☐ を ☐ ONする
☐ OFFする

○ 内部信号 ☐ を ☐ ONする
☐ OFFする

○ ボット入力 ☐ ボット入力に、16進2桁 ☐ とANDをとって、1 ☐ へ、ストアします。

○ ボット出力 ☐ ボットへ、1 ☐ の内容に、16進2桁 ☐ と、ANDをとって、出力します。
(ANDをとったBIT以外の変化はなし)

挿入 上書 削除

ラベル ☐ ヘジャンプし、それ以外は続行します。

【図23】

T	タスクの登録と開始
Act_Task n	別タスクの起動
Pause_Task n	別タスクの休止
Reset_Task n	別タスクのリセット
Stat?_Task n	別タスクが起動しているかチェック
S	自タスクから、次タスクへスイッチする
X	自タスクの自己終了と消滅

【図 2 4】

Go_Pclabel	無条件ジャンプ	
Gt_Pclabel	条件ジャンプ、 0 でない時ジャンプ	
Gf_Pclabel	条件ジャンプ、 0 の時ジャンプ	
Call_Pclabel	サブルーチン コール	
R	サブルーチン リターン	
Sel_c10, "1, Pc1, ... " 2, Pc2...	c10 が "1" なら Pc1、"2" なら Pc2 へジャンプ。それ以外は続行	
If_i10=i20,Pclabel If_i10<>i20,Pclabel If_i10>=i20,Pclabel If_i10<=i20,Pclabel If_i10>i20,Pclabel If_i10<i20,Pclabel If_i10=999,Pclabel If_i10<>999,Pclabel If_i10>=999,Pclabel If_i10<=999,Pclabel	If_i10>999,Pclabel If_i10<9999,Pclabel If_r10=r20,Pclabel If_r10<>r20,Pclabel If_r10>=r20,Pclabel If_r10<=r20,Pclabel If_r10>r20,Pclabel If_r10<r20,Pclabel	If (もしも) の条件に合えば Pclabel へジャンプ

【図 2 5】

I?_Isig	Isig 入力の ON/OFF をチェック	
O?_Osig	Osig 出力の ON/OFF をチェック	
V?_Vsig	Vsig 内部フラッグの ON/OFF をチェック	
On_ Osig	Osig 出力を ON する	
Of_ Osig	Osig 出力を OFF する	
Vn_ Vsig	Vsig 内部フラッグを ON する	
Vf_ Vsig	Vsig 内部フラッグを OFF する	
Pi_3,i10 & h'0f	入力 3 ポートと 16 進 0f と AND をとって i10 代入 (i10 の Hi バイトは 0 になる)	
Po_3,i10 & h'7f	i10 の Lo バイトと 16 進 7f と AND をとって出力 3 ポートへ出力する (出力 3 ポートの B7 は変化しない)	
Inw_Isig	Isig 入力の ON をウェイトする	
Inw_99,Pclabel, Isig	Isig 入力の ON をタイマー NO 99 の値の分だけウェイトし、タイムオーバーで Pclabel へジャンプし、時間内に ON すれば命令続行する	
Inw_C,Pclabel, Isig	以前の監視タイマーの続行で Isig 入力の ON をウェイト	
Ifw_Isig	Inw の反対で OFF をウェイトする	
Ifw_99,Pclabel, Isig		
Ifw_C,Pclabel, Isig		
Onw_ Osig	Ofw_ Osig	出力信号のウェイト
Onw_99,Pclabel,Osig	Ofw_99,Pclabel,Osig	
Onw_C,Pclabel, Osig	Ofw_C,Pclabel,Osig	
Vnw_ Vsig	Vfw_ Vsig	内部信号のウェイト
Vnw_99,Pclabel,Vsig	Vfw_99,Pclabel,Vsig	
Vnw_C,Pclabel,Vsig	Vfw_C,Pclabel,Vsig	

【図 26】

Tmset__9,99	タイマーNO 99の内容値を外部タイマー9にセットする
Tmup?__9	外部タイマー9が0か0でないかチェック
Dlyw__65535	タスク固有タイマーに 65535 データを直接セットしてディレイする
Dlywi__i 10	タスク固有タイマーに i 10 の内容値をセットしてディレイする
Dlywn__99	タスク固有タイマーにタイマーNO 99の内容値をセットしてディレイする

【図 27】

Ad__i 10, 7	i 10へ7chから A/D入力する (0-7)
Da__i 10, 1	i 10から1chへ D/A出力する (0,1)
\$ 0__c 10, 32	c 10 から 32 バイト送信する。送信完了まで、このアドレスでウェイトする (0CH 用)
\$ 1__c 10, 32	c 10 から 32 バイト送信する。送信完了まで、このアドレスでウェイトする (1CH 用)
% 0__c 10, 32	受信をウェイトし、受信完了でc 10 から 32 バイトセットする (0CH 用)
% 1__c 10, 32	受信をウェイトし、受信完了でc 10 から 32 バイトセットする (1CH 用)

【図 28-1】

i 10 = i 20 + i 30 i 10 = i 20 - i 30 i 10 = i 20 * i 30 i 10 = i 20 / i 30	正整数レジスタ i 同志の四則演算
i 10 = 65535 i 10 = i 20 + 65535 i 10 = i 20 - 65535 i 10 = i 20 * 65535 i 10 = i 20 / 65535	正整数レジスタ i と定数との四則演算
r 10 = r 20 + r 30 r 10 = r 20 - r 30 r 10 = r 20 * r 30 r 10 = r 20 / r 30	実数レジスタ r 同志の四則演算
i 10 = i 20 & i 30	正整数レジスタ i 同志の AND
i 10 = i 20 @ i 30	正整数レジスタ i 同志の OR
i 10 = i 20 & h' f 0 ff	正整数レジスタ i と定数との AND
i 10 = i 20 @ h' 00 ff	正整数レジスタ i と定数との OR
i 10 = R L (i 20, 15)	正整数レジスタ i 20 を 1 ビットずつ 15 回左ローテーション
i 10 = R R (i 20, 15)	正整数レジスタ i 20 を 1 ビットずつ 15 回右ローテーション
i 10 = S L (i 20, 15)	正整数レジスタ i 20 を 1 ビットずつ 15 回左シフト
i 10 = S R (i 20, 15)	正整数レジスタ i 20 を 1 ビットずつ 15 回右シフト
i 10 = D (i 20)	i 20 の内容を 10 進変換して i 10 にセット
i 10 = B (i 20)	i 20 の内容を 2 進変換して i 10 にセット
c 10 = "ABC..."	c 10 の先頭から、c 11、c 12、...と文字 ABC をセット
c 10 = h' 313233...	c 10 の先頭から、c 11、c 12、...と 16 進 31、32、33、にセット
c 10 = i 20 # S 5	i 20 の内容を c 10 から 5 文字分ゼロサプレス付きで、c 11、c 12、c 13、c 14、にセット
c 10 = i 20 # Z 5	i 20 の内容を c 10 から 5 文字分ゼロサプレス無しで、c 11、c 12、c 13、c 14、にセット

【図 28-2】

c 10= i 20# h	i 20 の 16 進内容をそのまま、c 10、c 11、c 12、c 13、にセット
c 10= r 20# S4.2	r 20 の内容を c 10 から整数部 4 桁、小数部 2 桁セット、ゼロサプレス付き
c 10= r 20# z3.3	r 20 の内容を c 10 から整数部 3 桁、小数部 3 桁セット、ゼロサプレス無し
i 10= c 20#5	c 20 から 5 文字分を i 10 に 2 進数としてセット
i 10= c 20#h	c 20、c 21、c 22、c 23、の 16 進表示をそのまま i 10 にセットする
r 10= c 20#6	c 20 から 6 文字分を r 10 に浮動小数点セット
i 10= r 20	r 20 整数部 4 桁を i 10 に 2 進数としてセット
r 10= i 20	i 20 の 5 桁 (0-65535) を r 10 にセット
r 10=SQRT (r 20)	平方根
r 10= (r 20) ** (r 30)	r 20 の r 30 乗
r 10=SIN (r 20)	S I N
r 10=COS (r 20)	C O S
r 10=ATN (r 20)	A R C T A N
r 10=LN (r 20)	自然対数
r 10=LOG (r 20)	常用対数

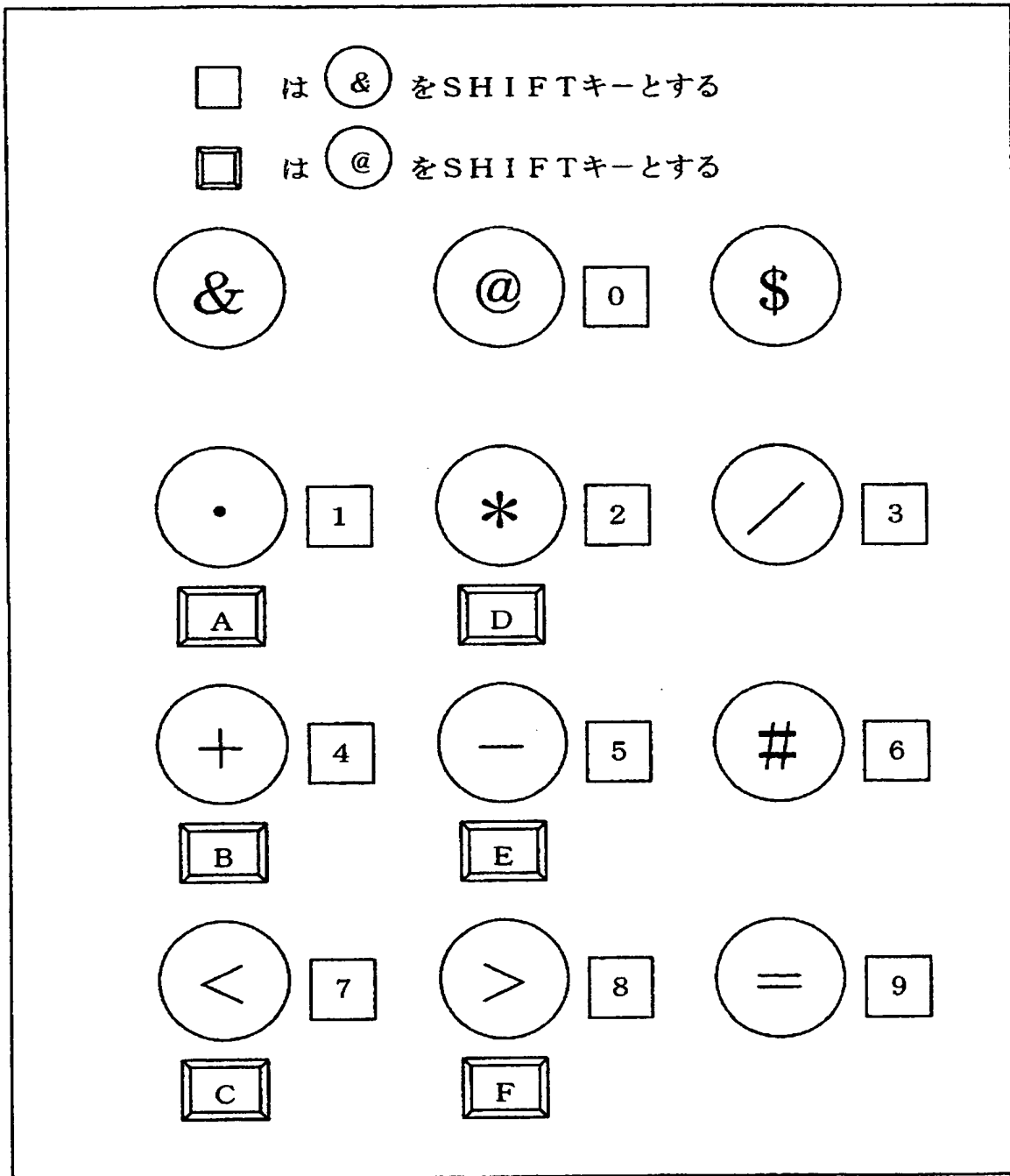
【図 29】

Ky__i 10	キー入力を即実行し、i 10 に 12 ビットセットする
Ky w__c 10	キー入力をウェイトし、押されればその変換キーコードを c 10 にセットする
Ch k?__c 10,"0","9"...	c 10 の内容が文字 0 から 9 のコード範囲かチェックする。範囲内なら F a l s e (0)
D c l r	画面クリア
D s p__3,5,"ABC	x=3, y=5 の位置から A B C 文字表示
D s p c__3,5,4,c 10	x=3, y=5 の位置から 4 文字分 c 10、c 11、c 12、c 13、の内容の文字表示
D r v s__3,5,5	x=3, y=5 の位置から 5 文字分反転文字
N u m w__3,5,4,c 10,d	x=3, y=5 の位置から 0~9 キーを入力表示しながら 4 文字分入力して ENT キー又はクリアキー押下までウェイトする。
N u m w__3,5,4,c 10,h	x=3, y=5 の位置から 0~9,A~F キーを入力表示しながら 4 文字入力して ENT キー又はクリアキー押下までウェイトする。
R v s s e t__i 10,3,5,4,...	x=3, y=5 の位置から 4 文字分、他 x,y,length をセットした分、i 10 をポインターとして、反転をループ状に行う、初期セット。
R v s i n c__i 10,3,5,4,...	上記 Rvsset と同じ設定で、この命令を実行する毎に反転がループ状に移行する。

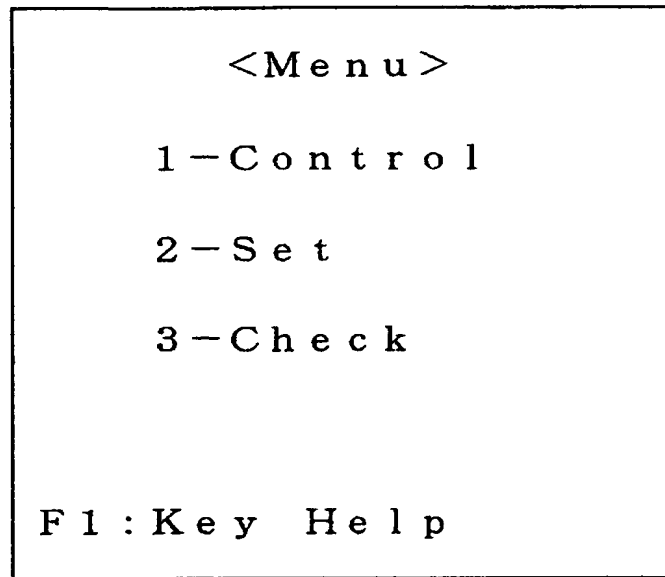
【図 30】

Strani_Vsig, I sig,0 Strani_Vsig, I sig,1 Strano_Osig, Vsig,0 Strano_Osig, Vsig,1	I sig をそのまま Vsig にセット I sig を反転して Vsig にセット Vsig をそのまま Osig にセット Vsig を反転して Osig にセット
Sand_Vsig1,0,Vsig2,0,Vsig3,0	Vsig 2 をそのままにしたものと Vsig 3 をそのままにしたものと AND をとってそのままにして Vsig1 にセッ ト
Sand_Vsig1,1,Vsig2,1,Vsig3,1	Vsig 2 を反転したものと Vsig 3 を反転したものと AND をとって反転して Vsig1 にセット
Sor_Vsig1,0,Vsig2,0,Vsig3,0	Vsig 2 をそのままにしたものと Vsig 3 をそのままにしたものと OR をとってそのままにして Vsig1 にセット
Sor_Vsig1,1,Vsig2,1,Vsig3,1	Vsig 2 を反転したものと Vsig 3 を反転したものと OR をとって反転して Vsig1 にセット
Db_AF,CD,56...	バイトテーブル
Dw_3457,AFFF...	ワードテーブル
Getc_Pclabel, c 10,5, i 20	横 5 バイト毎のラベル Pclabel バイトテー ブルから i 20 の内容値をインデックスとした、 横 1 列 5 バイトを c 10 から 5 文字分セット する。
Geti_Pclabel, i 10,5, i 20	横 5 ワード毎のラベル Pclabel ワードテー ブルから i 20 の内容値をインデックスとした、 横 1 列 5 ワードを i 10 から 5 ワード分セッ トする。
Sfc_c 10, 5, c 20	c 10 から 5 文字分を 1 文字ずつシフトして 先頭 c 10 に c 20 をセットする。
Sfi_i 10, 5, i 20	i 10 から 5 ワード分、1 ワードずつシフトし て先頭 i 10 に i 20 をセットする。
Mv_i 10,Data(i 20)	i 20 の内容をインデックスとして Data メモリーから i 10 へセットする。
Mv_Data(i 10), i 20	i 10 の内容をインデックスとして i 20 の内 容を Data メモリーへセットする。
Clock_c 10	カレンダークロックから読み出し c 10 を先 頭として 12 文字分、年月日時分秒セット (書き込みはコントローラーで出来る)
;	コメント。CR か ; まで。
:	ラベルを識別する。
E	プログラムの終了。

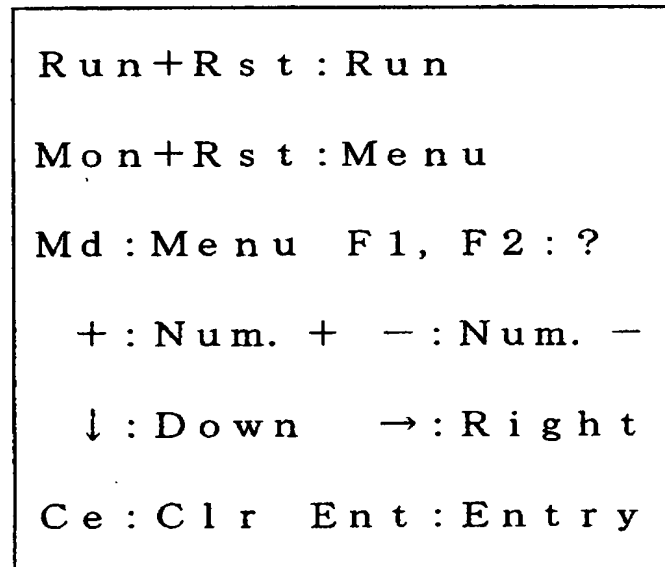
【図 31】



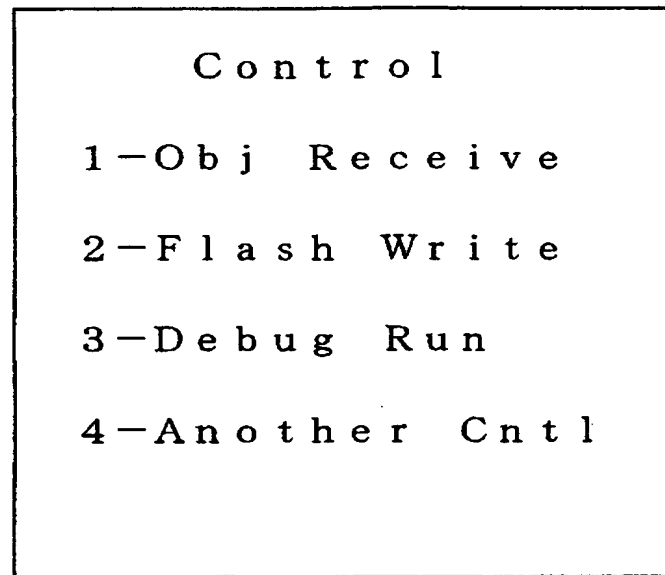
【図 32】



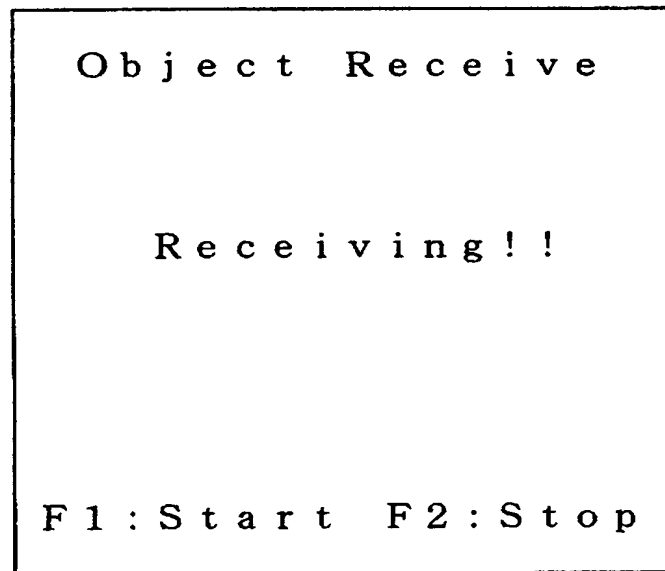
【図 33】



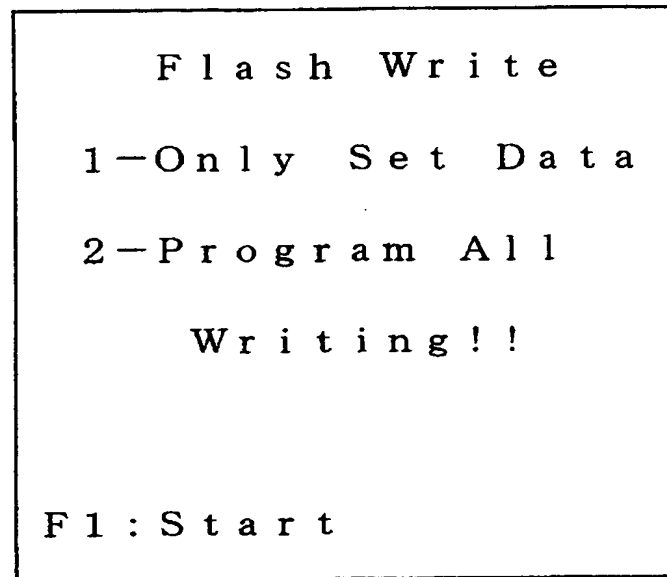
【図34】



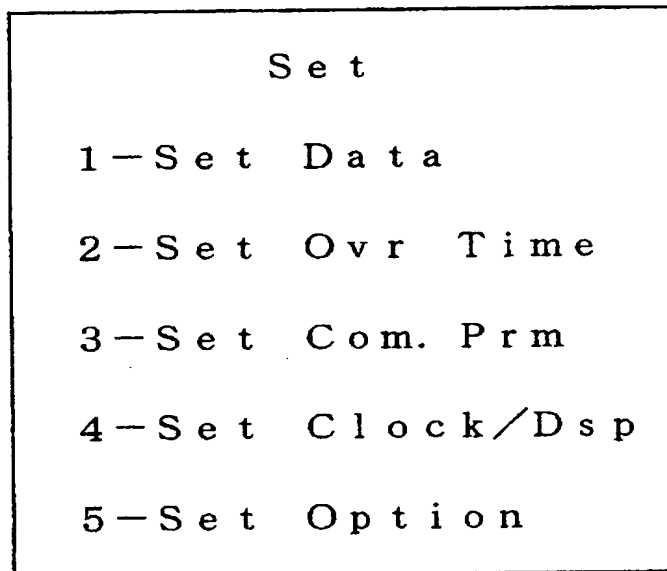
【図35】



【図 36】



【図 37】



【図 38】

S e t D a t a	
N o	D a t a
D 5 0	: 1 0 0 0
F 1 : N o F 2 : D a t a	

【図 39】

S e t O v e r T i m e	
N o	D a t a
9 9	: 1 0 0 m s
F 1 : N o F 2 : D a t a	

【図40】

Como Parameter
Baud Rate : 38400
(48, 96, 192, 384)
Bit : 8 (7, 8)
Stop : 1 (1, 2)
Check : N (N, E, O)

【図41】

Com1 Parameter
Baud Rate : 38400
(48, 96, 192, 384)
Bit : 8 (7, 8)
Stop : 1 (1, 2)
Check : N (N, E, O)

【図 4 2】

Com. Terminator
 0No (No, Cr, Lf, Ex)
 1No (No, Cr, Lf, Ex)
 Com. Chr Ovr Time
 0: 3ms 1: 3ms
 F1: Num Set Mode

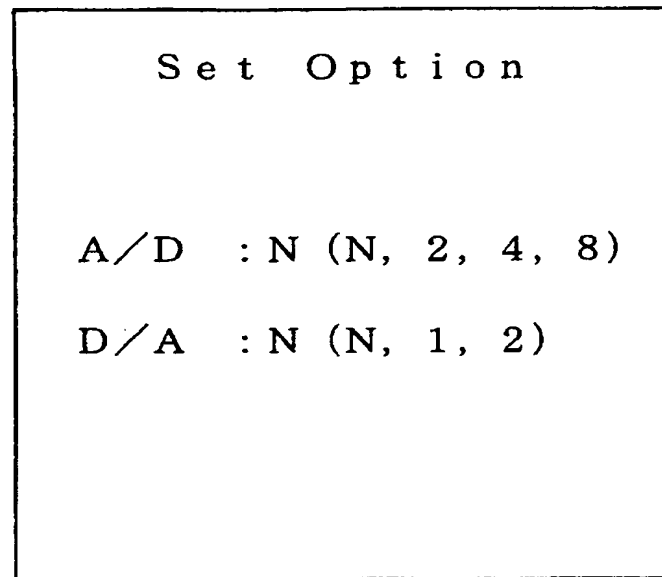
【図 4 3】

Clock

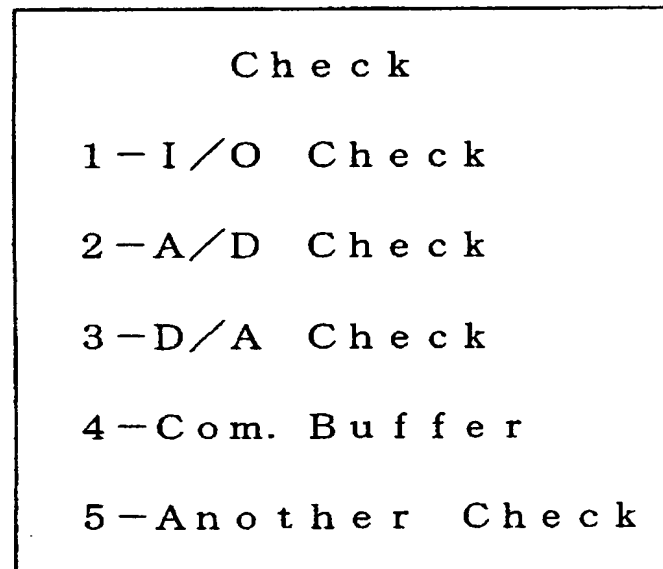
 02Y12M31D
 13H04M05S

 F1: Write Mode

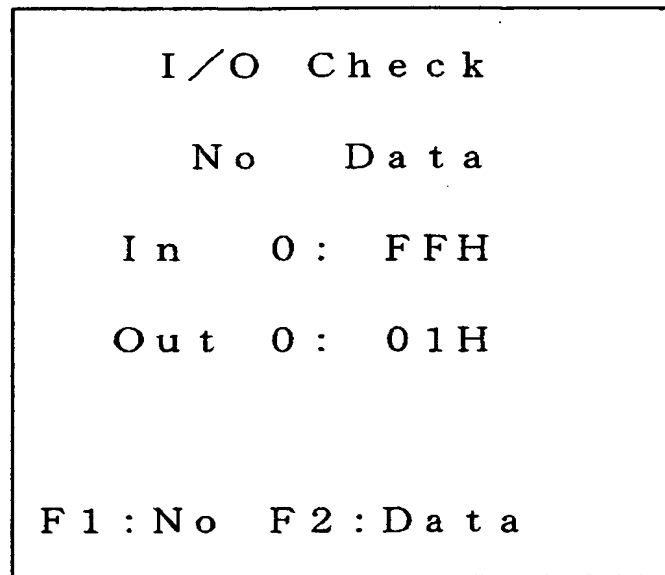
【図 44】



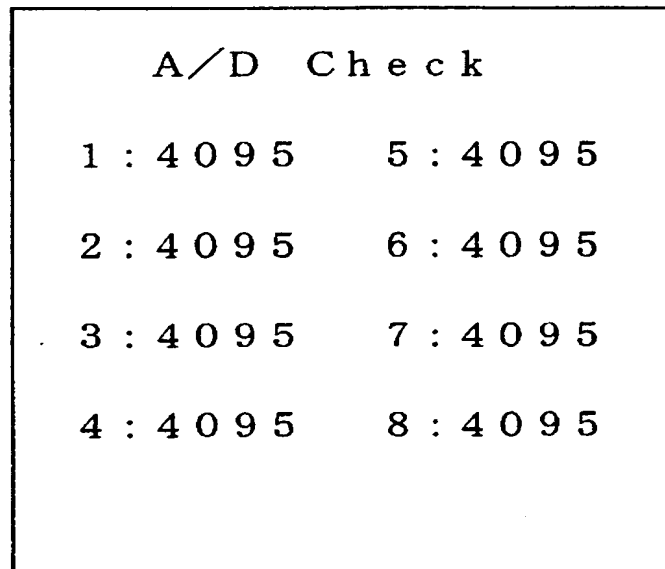
【図 45】



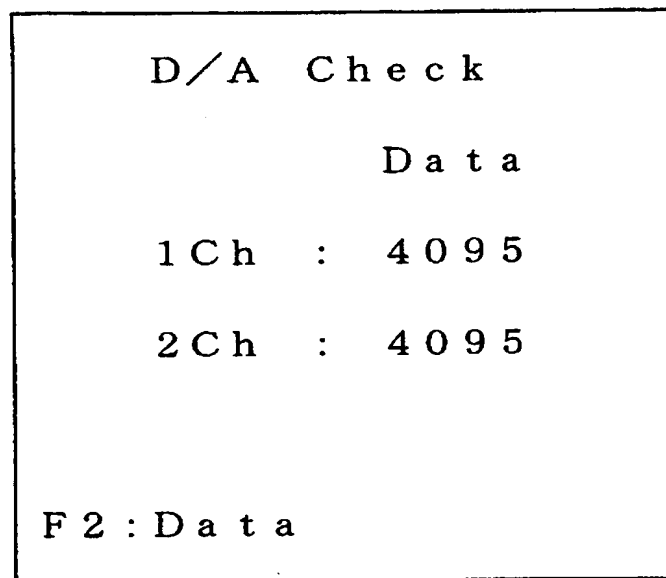
【図 46】



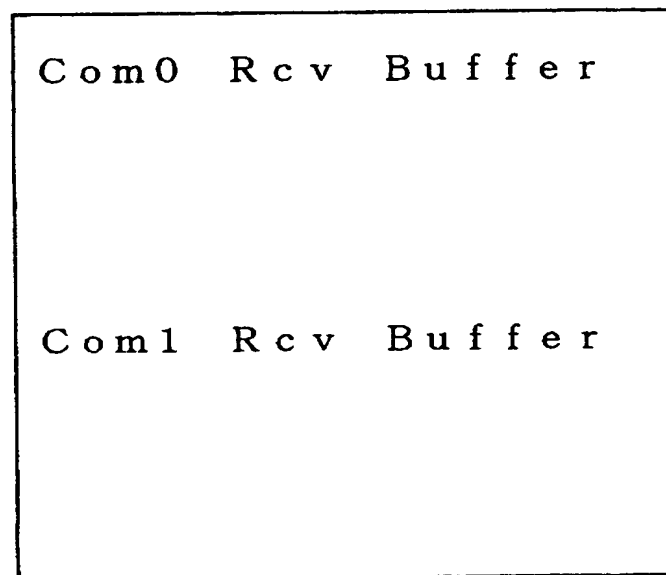
【図 47】



【図 48】



【図 49】



【図 50】

A n o t h e r C h e c k	
C y c l e T	7 m s
I 00 : 65535	C 00 : 31H
R 00 : +0.123456-03	
V f f : O N	
F 1 : N o	

【書類名】 要約書

【要約】

【課題】

コンピューター技術、特に産業機器は特異な言語を駆使するため技術者には一般大衆と隔絶した砦の様なものが形成しており、時には、それが弊害となり、意思の疎通が感じられる事もあり、しばしばその横暴ぶりに痛感する事がある。

制作金額、納期について、我を通し、簡単な変更の発生時にも価格的に高目になったり、しばしばユーザーの嘆きを聞く事になる。又、プログラマーとして起業した初心者に関して、第 1 回の開発でつまずき、二度と信頼を得る事がなく、挫折の浮き目に会った多くの人々を見てきた。

本、発明はその様な砦を壊し、誰でも、気安く F A に従事して行ける様、一念発起して制作したものである。

【解決手段】

多重処理を意識させずに、流れ図（フローチャート）にそった形で、しかも、プログラム制作に指針を与えながら選択式にして、コーディング作業などの開発工数を省く。パッケージ化する事により、バグ発生をさせないようにする。

認定・付加情報

特許出願の番号	特願 2002-383083
受付番号	20202330387
書類名	特許願
担当官	塩野 実 2151
作成日	平成15年 4月 8日

<認定情報・付加情報>

【提出日】	平成14年12月 6日
【特許出願人】	申請人
【識別番号】	503029670
【住所又は居所】	大阪府大阪市鶴見区鶴見 5丁目 6番 9号 301室
【氏名又は名称】	加藤 宝一

次頁無

特願 2 0 0 2 - 3 8 3 0 8 3

出 願 人 履 歴 情 報

識別番号

[5 0 3 0 2 9 6 7 0]

1 . 変更年月日

2 0 0 2 年 1 2 月 6 日

[変更理由]

新規登録

住 所

大阪府大阪市鶴見区鶴見 5 丁目 6 番 9 号 3 0 1 室

氏 名

加藤 宝一